Prpall项目开发规范

2015年9月

目录

[1 前言 3](#_Toc429770438)

[1.1 文档编写目的 3](#_Toc429770439)

[1.2 文档适用范围 3](#_Toc429770440)

[1.3 修订历史 3](#_Toc429770441)

[2 开发环境和配置 4](#_Toc429770442)

[2.1 开发规范总揽 4](#_Toc429770443)

[2.1.1 模块分解图 4](#_Toc429770444)

[2.1.2 模块简要说明 4](#_Toc429770445)

[2.2 工程目录规范 5](#_Toc429770446)

[2.2.1 JSP目录规范 5](#_Toc429770447)

[2.2.2 WEB目录规范 6](#_Toc429770448)

[2.2.3 JAVA目录规范 6](#_Toc429770449)

[2.2.4 RESOURCE目录规范 7](#_Toc429770450)

[2.3 JSP/HTML开发规范 8](#_Toc429770451)

[2.3.1 JSP流转图示 8](#_Toc429770452)

[2.3.2 命名规范 8](#_Toc429770453)

[2.3.3 编写规范 9](#_Toc429770462)

[2.3.4 注释规范 10](#_Toc429770463)

[2.4 JAVASCRIPT开发规范 11](#_Toc429770464)

[2.4.1 Javascript库维护规则 11](#_Toc429770465)

[2.4.2 命名规范 11](#_Toc429770466)

[2.4.3 编写规范 12](#_Toc429770467)

[2.4.4 注释规范 13](#_Toc429770468)

[2.5 JAVA开发规范 14](#_Toc429770469)

[2.5.1 命名规范 14](#_Toc429770470)

[2.5.2 编写规范 15](#_Toc429770471)

[2.5.3 JAVA语言规则 16](#_Toc429770484)

[2.5.4 注释规范 17](#_Toc429770485)

[2.6 配置文件开发规范 18](#_Toc429770486)

[2.6.1 命名规范 18](#_Toc429770487)

[2.6.2 编写规范 18](#_Toc429770488)

[2.6.3 注释规范 18](#_Toc429770489)

# 前言

本文档是《prpall项目开发规范》，文档讲述的内容要求所有开发人员必须遵守。

开发人员还需要查阅《prpall项目开发指南》，《prpall项目开发规范》和《prpall项目开发指南》具有同等效力。

参与项目开发的任何开发人员如果对本文档讲述的内容有疑问，在开发前请询问项目经理，不允许任何开发人员以任何理由违反本文档讲述的规范要求。

## 文档编写目的

规范开发思想，约束开发行为，明确开发纪律。

特别约定：

1. 本章规范中，我们用“module”代表“模块”。例如“module的POJO类”表述为“模块的POJO类”。
2. 本章规范中所有的英文单词都是严格遵守大小写的。

## 文档适用范围

本文档适用于使用prpall项目的项目组成员。下表是本文档对于项目组成员的最低要求程度：

|  |  |  |
| --- | --- | --- |
| 项目组成员 | 是否必读 | 要求程度 |
| 项目经理 | ■是 □否 | □了解文档结构 ■明白文档内容 □透彻理解 |
| 系统分析师（业务） | ■是 □否 | ■了解文档结构 □明白文档内容 □透彻理解 |
| 系统分析师（技术） | ■是 □否 | □了解文档结构 □明白文档内容 ■透彻理解 |
| 高级程序员 | ■是 □否 | □了解文档结构 ■明白文档内容 □透彻理解 |
| 开发工程师 | ■是 □否 | □了解文档结构 ■明白文档内容 □透彻理解 |
| 质量保证工程师 | □是 ■否 | □了解文档结构 □明白文档内容 □透彻理解 |

对于项目经理、系统分析师、高级程序员和开发工程师，还需要阅读《prpall项目架构设计说明书》和《prpall项目开发指南》。

## 修订历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修订章节 | 类型 | 日期 | 作者 | 说明 |
| 全部 | 创建 | 2015-09-11 | 张百成 | 新建文档 |
| 全部 | 修改 | 2015-09-11 | 张百成 | 修正不足，细化内容 |

# 开发环境和配置

必须遵守《prpall开发环境搭建指南》的讲述。

## 开发规范总揽

### 模块分解图



### 模块简要说明

下面对每一个模块需要描述的命名规范进行简要的说明：

* 前台页面：描述前台页面文件的命名规则、前台action请求的命名规则。
* Spring配置文件：描述Spring对于Controller和Service的bean命名规范。
* Java类：描述每一个功能对应的Controller类、Service类和Service实现类的命名规范。
* Java类中的方法：描述Controller类中增删改查方法和其他业务方法的命名规范，以及Service类中方法的命名规范。
* Java类中的变量：描述Controller类中Service变量和各辅助变量的命名规范。

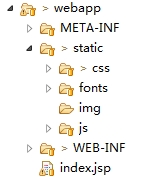
具体参看以下内容。

## 工程目录规范

以我们实际工程为例，我们主要分为3个部分，JSP页面，resource配置文件和java类文件。这三个方面构成了我们整体的工程体系，也是我们日常接触最多的目录体系。所以说一个规范的目录体系是我们赖以开发的基础和模板。

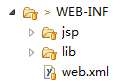
除特别用途外，目录名必须全部为小写。

### 根目录规范



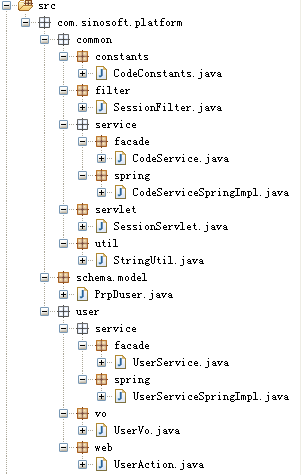
|  |  |
| --- | --- |
| **目录** | **说明** |
| static | 静态文件目录 |
| static/css | 样式文件目录 |
| static/fonts | 字体文件目录 |
| static/img | 图片文件目录 |
| static/js | JS文件目录 |
| WEB-INF | Web应用系统部分，包括配置文件、库文件、类文件等 |

### WEB目录规范



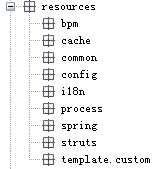
|  |  |
| --- | --- |
| **目录** | **说明** |
| WEB-INF | Web应用系统部分，包括配置文件、库文件、类文件等 |
| WEB-INF/classes | 类文件目录 |
| WEB-INF/jsp | Jsp文件目录 |
| WEB-INF/lib | Jar文件目录 |

### JAVA目录规范



|  |  |
| --- | --- |
| **包名** | **说明** |
| com.prpall | prpall包目录 |
| com.prpall.controller | SpringMVC的Controller目录 |
| com.sinosoft.platform.common | SpringMVC的Controller目录 |
| com.sinosoft.platform.common.constants | 公共常量静态类目录 |
| com.sinosoft.platform.common.filter | 公共过滤器目录 |
| com.sinosoft.platform.common.service | 公共服务目录 |
| com.sinosoft.platform.common.service.facade | 公共服务接口目录 |
| com.sinosoft.platform.common.service.spring | 公共服务Spring方式实现目录 |
| com.sinosoft.platform.common.servlet | 公共servlet类目录 |
| com.sinosoft.platform.common.util | 公共工具类目录 |
| com.sinosoft.platform.schema.model | POJO对象目录（ORM的POJO） |
| com.sinosoft.platform.user | User模块根目录 |
| com.sinosoft.platform.user.service | User服务目录 |
| com.sinosoft.platform.user.service.facade | User服务接口目录 |
| com.sinosoft.platform.user.service.spring | User服务Spring方式实现目录 |
| com.sinosoft.platform.user.vo | User自定义VO目录 |
| com.sinosoft.platform.user.web | Struts2方式的Action目录 |

### RESOURCE目录规范



|  |  |
| --- | --- |
| **包名** | **说明** |
| resources | 配置文件根目录 |
| resources/bpm | 工作流jbpm配置文件目录 |
| resources/cache | cache配置文件目录 |
| resources/common | 公共配置文件目录 |
| resources/config | 工程properties配置文件目录 |
| resources/i18n | 国际化配置文件目录 |
| resources/process | 工作流配置模板文件目录 |
| resources/spring | spring配置文件目录 |
| resources/struts | struts配置文件目录 |
| resources/template/custom | Struts自定义解析模板文件目录 |

## JSP/HTML开发规范

### JSP流转图示



### 命名规范

除特别用途外，文件名区分大小写，每个单词的首字母大写。

作为jsp我们可以允许两种命名规范。

#### 对象+操作



##### 规则

规则为：对象＋操作

|  |  |  |  |
| --- | --- | --- | --- |
| **操作名** | **适用文件类型** | **含义** | **备注** |
| Edit | JSP | 编辑 | 可以包含View的功能，通过页面逻辑控制 |
| View | JSP | 显示 | 也可以在Edit页面中进行维护 |
| List | JSP | 列表 | 列表查询页面 |

以对象User为例，其文件名称如下所示：

|  |  |
| --- | --- |
| **文件名** | **含义** |
| UserEdit.jsp | 编辑 |
| UserView.jsp | 显示 |
| UserList.jsp | 列表 |

##### 适用范围

此类命名规范适用于类似平台系统中的单表操作模块命名，一个模块即可通过制定的几个JSP文件完成的情况。例如User模块，即可UserList.jsp；UserEdit.jsp；UserView.jsp即可描述User这个模块的增删改差功能。



#### 模块+操作

##### 规则

规则为：模块＋操作

例如：CheckMainEdit.jsp

CheckMain作为模块名+操作Edit组成jsp命名。

|  |  |  |  |
| --- | --- | --- | --- |
| **操作名** | **适用文件类型** | **含义** | **备注** |
| Edit | JSP | 编辑 | 可以包含View的功能，通过页面逻辑控制 |
| View | JSP | 显示 | 也可以在Edit页面中进行维护 |
| List | JSP | 列表 | 列表查询页面 |

|  |  |  |  |
| --- | --- | --- | --- |
| **模块名** | **操作名** | **文件名** | **含义** |
| CheckMain | Edit | CheckMainEdit.jsp | 查勘主信息编辑 |
| CheckCarInfo | View | CheckCarInfoView.jsp | 查勘车辆信息查看 |
| ClaimBusiness | List | ClaimBusinessList.jsp | 立案业务列表查询 |

##### 适用范围

适用于核心业务系统，如理赔，承保等等。核心业务系统不是单表，所以第一种的命名方式并不能满足我们的规范需求，也不能起到文件清晰处理的效果。所以此处我们同样也可以使用模块名+操作名的方式来命名。操作名同样是我们的Edit，View，List就可以满足，而具体的模块名还请大家使用具体业务环境的词根表进行规范命名。

### 编写规范

1. 所有JSP文件的第一行必须写为：<%@ page contentType="text/html; charset=GBK"%>。
2. JSP文件中的中文注释必须写在“<%-- 中文注释 --%>”内。
3. 所有的HTML标签（例如<html>、<body>、<table>、<tr>、<td>等）都必须是小写字母。
4. <html>、<head>、<body>标签必须顶行写，在它之前不允许有空格。其他所有的元素标签依照嵌套关系依次所进两个空格。
5. 元素的开始标签和结束标签必须成对出现（例如<table>和</table>），且必须在同一列上。
6. 如果某个元素标签的属性太多需要换行写，新起行的第一个属性的列数必须在这个元素的开始标签的列数之后。
7. JSP/HTML文件中的TAB键必须替换成空格，且依次所进两个空格。
8. JSP/HTML文件每行字符为120个包括空格，超过的部分要换行写。
9. 不允许在JSP中编写JS代码，需要另建JS文件引入使用。

### 注释规范

经常遇到的JSP页面注释方式包括以下两种:

1. 用户可以看到的注释;符号:<!-- -->.用户可以看到指的是用户可以通过浏览器的“查看-源文件”看到注释的内容。(并不是直接显示在页面上)。
2. 用户看不到的注释符号：<%-- --%>或者:<%/\*\* 内容\*\*/%>。这类注释主要是面向程序员的注释.方便程序员了解和更新程序代码。
3. 用途: 方便程序员进行jsp页面的调试。一般用<!-- -->来进行调试，因为你可以在注释内部加入EL（表达式），从而到运行页面点击浏览器：查看-源代码查看表达式的结果。用<%-- --%>或者<%/\*\*\*\*/%>不能达到这个目的，因为这个注释下的内容无法在浏览器中查看。

例如：

<html>  
<head>  
<title>tests</title>

<%  
  String name = "shuxudong";  
%>  
</head>  
<h1>Quoting test</h1>  
<!--这个注释用户可以看到[<%=name%>] -->  
<%--这个注释用户看不到[<%=name%>] --%>  
</html>

## JAVASCRIPT开发规范

### Javascript库维护规则

软件的长期价值直接源于其编码质量。在它的整个生命周期里,一个程序可能会被许多人阅读或修改。如果一个程序可以清晰的展现出它的结构和特征,那就能减少在以后对其进行修改时出错的可能性。编程规范可以帮助程序员们增加程序的健壮性。

所有的JavaScript代码都是暴露给公众的。所以我们更应该保证其质量。保持整洁很重要。

### 命名规范

JavaScript 程序应独立保存在后缀名为 .js 的文件中。

#### JS文件名命名

例如一个叫做UserEdit的jsp文件中需要js方法，我们会新建一个叫做UserEdit.js的文件区编写相关的JS方法，而不是在jsp中直接编写。

#### 模块命名

在同一个模块下的js可以可以公共一个js文件，比如在user的模块中，我们可以建立一个叫做User.js来进行共用。

### 编写规范

1. JavaScript文件

JavaScript 代码不应该被包含在 HTML 文件中，除非这是段特定只属于此部分的代码。在 HTML 中的JavaScript代码会明显增加文件大小，而且也不能对其进行缓存和压缩。

filename.js 应尽量放到 body 的后面。这样可以减少因为载入脚本而造成其他页面内容载入也被延迟的问题。

1. 变量名命名规范

\* 常量及全局变量采用全部大写的形式。

\* 变量名采用大小写混合形式，除首单词的字母全部小写外其余单词的首字母都大写。

\* 变量名过长的可以采用单词缩写，除广泛使用的缩写外，采用单词缩写的变量名必须添加注释。

1. 变量使用规范

\* 变量名使用之前必须定义，未定义的变量名禁止使用。

\* 变量的使用要尽量的减小作用域。

不建议的使用方式：

     var I;

      for(I=0;I<MAX\_COUNT;I++)

     {

      }

     应该这样使用

    for(var I = 0;I<MAX\_COUNT;I++)

1. 函数及子名称的命名规范

\* 函数名尽量采用 “动词+名次”的命名方式。

\* 函数名称中的动词前缀必须是和函数功能相关的完整动词。

\* 函数名称的第一个单词全部小写，后面的单词首字母大写。

\* 函数的大括号要垂直对齐，例如：

function getUserName()

     {

       .........//此处要开始缩进。

     }

1. 缩进

缩进的单位为四个空格。避免使用 Tab 键来缩进。因为始终没有个统一的 Tab 长短标准。虽然使用空格会增加文件的大小，但在局域网中几乎可以忽略，且在最小化过程中也可被消除掉。

1. 函数声名

所有的函数在使用前进行声明。 内函数的声明跟在var语句的后面。这样可以帮助判断哪些变量是在函数范围内的。

函数名与“(”（左括号）之间不应该有空格。“)”（右括号）与开始程序体的“{”（左大括号）之间应插入一个空格。函数程序体应缩进四个空格。“}”（右大括号）与声明函数的那一行代码头部对齐。

function outer(c，d) {  
var e = c \* d;  
  
function inner(a， b) {  
return (e \* a) + b;  
}  
  
return inner(0， 1);  
}

### 注释规范

单行注释

插入单行注释的符号是“//”

如：<script type=” text/javascript”>

//javascript单行注释

Alert(“这是javascript单行注释”);

</script>

2．多行注释

多行注释以"/\*"开始，以"\*/"结束。

如：<script type=” text/javascript”>

/\*

Javascript

多行注释

Alert(“这是javascript多行注释”);

\*/

Alert(“这是javascript”);

</script>

原则上，所有的Javascript脚本都应该兼容Microsoft Internet Explorer 6.0和Netscape Navigator 7.以上版本。

## JAVA开发规范

### 命名规范

此处的java文件已经是我们工程一部分了，所以java的命名也就是需要满足目录规范的命名。

#### Model命名

我们的model对象全部由Hibernate Tools生成，也就是我们的model对象需要和数据表进行映射.

按Java规则，首字母大写，之后的每个单词的首字母也大写。

注意：当表名为PrpLXXX（或PrpCXXX、PrpTxxx）时， Prp作为一个单词，LXXX作为一个单词

如表名为prpduser时类名为PrpDuser

例如：

|  |  |
| --- | --- |
| **对象名（表名）** | **类名** |
| prpduser | PrpDuser |
| prplregist | PrpLregist |
| prpldeflossmain | PrpLdeflossMain |

#### Web层Action的命名

1. Action类必须存放在com.sinosoft.subsystems.module.web包下。
2. com.sinosoft.subsystems.module.web包下的类全部都要以“Action”结尾。
3. Action类必须继承ins.framework.web.Struts2Action类。
4. 类名为：模块名+后缀，如user模块的Web层Java类名为 UserAction

#### Service层命名

1. com.sinosoft.subsystems.module.service包下不能有文件。
2. com.sinosoft.subsystems.module.service.facade包下的接口名必须以Service结尾。
3. com.sinosoft.subsystems.module.service.spring包下的类名必须以ServiceSpringImpl结尾。
4. 类名为：模块名+后缀，如user模块的业务层Java类的facade名为 UserService，其Spring的实现类的名字为UserSeviceSpringImpl

### 编写规范

#### Action编写规范



##### 类编程规范

1. private final Logger logger = Logger.getLogger(UserAction.class);方法的入参必须是该Action类本身的class对象。
2. 在WEB层类不允许用“new”关键字创建业务层和持久层类对象。
3. 在WEB层类不允许出现与SQL语句相关的代码，不允许调用任何类拼写SQL语句。
4. 定义需要的变量和注入的业务服务应建立set，get方法。

##### 方法编程规范

1. 调用业务层的Service必须通过接口注入。
2. 在查询和翻页查询的方法中，必须把从业务层Service返回的结果集存放到Page对象中。
3. WEB层的类之间不允许互相调用。
4. WEB层仅可以抛出DataVerifyException和PermissionException。

方法名为 动词＋名词，如新增user方法名为 addUser，更新user为updateUser。特定模块处理也可以就直接使用add作为方法名，不过并不推荐。方法名最好要有具体的含义以便方便大家理解。



#### Service编写规范

##### 类编程规范

1. 所有的ServiceSpringImpl类必须实现对应的Service接口，类和接口的名称仅相差“SpringImpl”。
2. ServiceSpringImpl类必须配置在/src/resources/spring/ applicationContext-service.xml文件中，<bean>的id属性值必须是首字母小写的Service接口名。
3. ServiceSpringImpl类的类变量只能用于定义Service引用，且只可用private修饰，对于这些Service引用，只需要在ServiceSpringImpl类中生成set方法。
4. ServiceSpringImpl类的Service引用的变量名必须与applicationContext-service.xml文件中定义的<bean>节点的id属性值相同。
5. 在ServiceSpringImpl类中不允许使用“new”关键字创建持久层对象和WEB层对象。
6. 在业务层仅可以抛出BusinessException。
7. ServiceSpringImpl类必须继承GenericDaoHibernate类，并完成填写泛型定义。

##### 方法编程规范

Service方法命名规范如下：

|  |  |  |
| --- | --- | --- |
| **方法前缀** | **说明** | **示例** |
| add | 增加 | addUser |
| update | 更新 | updateUser |
| find | 查询、获取等(单条或多条记录） | findUser、findUserByUserCode |
| delete | 删除 | deleteUser |

注意：由于和getter方法混淆，这里不采用get作为业务方法的前缀，而改用find。

### JAVA语言规则

具体的java语言规范还可以通过PMD和checkstyle等等工具进行相关的检查，附件中师PMD的检查项。



### 注释规范

1. 单行注释

//此处为单行注释

1. 多行注释

/\*

\* 此处为多行注释

\* 注意使用方式

\*/

1. 变量注释

每定义一个变量或者引入一个方法都应该增加相关的注释。

1. 方法注释

每一个方法都应该写完成的方法注释以便作为服务以供系统使用。

/\*\*

\* 翻译代码<br>

\* 支持的代码类型有：<br>

\* <pre>

\* UserCode 员工代码

\* ComCode 机构代码

\* </pre>

\* 例如以下代码，查询性别代码为1的中文名称

\* <pre>

\* String value = codeService.translateCode(&quot;SexCode&quot;, &quot;1&quot;, &quot;DAA&quot;, &quot;C&quot;);

\* </pre>

\* @param codeType

\* 代码类型

\* @param codeCode

\* 代码

\* @param riskCode

\* 险种代码

\* @param language

\* 语种(C:中文/E:英文)

\* @return 代码名称

\*/

public String translateCode(String codeType, String codeCode,

String riskCode, String language);

## 配置文件开发规范

### 命名规范

在Arch4的框架中我们还需要对配置文件进行日常的维护。主要涉及Struts和Spring的部分配置文件。

|  |  |  |  |
| --- | --- | --- | --- |
| **配置文件** | **命名规范** | **说明** | **示例** |
| Struts | struts-\*.xml | 具体模块的Struts配置 | 1.struts-user.xml员工管理配置  2.struts-check.xml查勘配置 |
| Spring | applicationContext-\*.xml | 具体功能的spring配置 | 1.applicationContext-action.xml  spring的action配置  2. applicationContext-jms.xml  Spring的jms配置 |

如有需要可以自己新定义相关的配置文件，不过不推荐。特殊需要请联系项目管理人员。

### 编写规范

具体规范请结合开发指南。

### 注释规范

Xml注释请使用

<!-- 此处写明需要的注释内容 -->